Expert Systems with Applications 38 (2011) 11072–11077

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A hybrid ant colony optimization for continuous domains

Jing Xiao [a,*], LiangPing Li [b]

[a] School of Computer Science, South China Normal University, No. 55 Zhongshan West Road, Guangzhou 510631, China
[b] Department of Computer Science, Sun Yat-sen University, No. 132 WaiHuan East Rd., Guangzhou Higher Education Mega Center, Guangzhou 510006, China

ARTICLE INFO

Keywords:
Continuous optimization
Ant colony optimization
Continuous population-based incremental learning
Differential evolution

ABSTRACT

Research on optimization in continuous domains gains much of focus in swarm computation recently. A hybrid ant colony optimization approach which combines with the continuous population-based incremental learning and the differential evolution for continuous domains is proposed in this paper. It utilizes the ant population distribution and combines the continuous population-based incremental learning to dynamically generate the Gaussian probability density functions during evolution. To alleviate the less diversity problem in traditional population-based ant colony algorithms, differential evolution is employed to calculate Gaussian mean values for the next generation in the proposed method. Experimental results on a large set of test functions show that the new approach is promising and performs better than most of the state-of-the-art ACO algorithms do in continuous domains.

1. Introduction

Inspired by the ants' foraging behavior, the first ant colony algorithm was proposed in (Dorigo, 1992). The core idea of an ant system is based on the fact that each ant deposits pheromone on the foraging path. In the early research, ant colony algorithms have been successfully applied for solving combinatorial optimization problems, including the traveling salesman problem (TSP) (Dorigo & Gambardella, 1997), the quadratic assignment problem (QAP) (Maniezzo, Colorni, & Dorigo, 1999) and the job shop scheduling problem (JSP) (Colorni, Dorigo, Maniezzo, & Trubian, 1994).

The extension of the original ant colony algorithm to continuous domain can be accomplished by either discretizing the continuous domain into several regions or shifting from using a discrete probability distribution to using a continuous one such as a Gaussian probability density function (*pdf*). The first extension of ant colony algorithms to continuous domain is CACO (continuous ant colony optimization) (Bilchev & Parmee, 1995). CACO discretized the continuous neighborhood with nest structure. It initialized a nest on a given point of the search space and generates random vectors. The direction vector chosen was updated by the ants with better fitness values. In API (ant pachycondyla apicalis) (Monmarché, Venturini, & Slimane, 2000) each ant searched independently for a solution and the ants' nest is periodically moved. CIAC (continuous interacting ant colony) (Dréo & Siarry, 2002) used some attractive points on the search space and direct communication between ants to improve the exploration. COAC (continuous orthogonal ant colony) (Hu, Zhang, & Li, 2008) also discretized the continuous search space and utilized the orthogonal design method to search the continuous domain completely.

Another ensemble of ant colony optimization algorithms for continuous domain is mainly based on Gaussian probability density function sampling. CACS (continuous ant colony system) (Pourtakdoust & Nobahari, 2004) employed a Gaussian probability density function whose mean and variance are adjusted during evolution to sample the continuous search space. MACACO (multivariate ant colony algorithm for continuous optimization) (Franca, Coelho, Von Zuben, & de Faissol Attux RR, 2008) optimized the search space with multivariate Gaussian pdf which is created by the information contained in the covariance matrix of the ant population. Speak of population, there are some work which fully utilize the information in previous ant population to generate the next one. PB-ACO (population-based ACO) (Guntsch & Middendorf, 2002) kept track of a certain number of best solutions found so far and used the archive to update the pheromone. PB-ACO was applied to the TSP and QAP problems while ACOR (ant colony optimization in Rn) (Socha & Dorigo, 2008) extended the population-based ACO with Gaussian *pdf* as pheromone update. ACO_R evolved several Gaussian *pdfs* in parallel and adopted the rank-based selection mechanism. ACO_R also allowed exploiting the correlation between variables by coordinate rotation which is time-consuming.

Inspired by PB-ACO and ACO_R, our motivation for introducing a population based scheme is the dynamic generation of probability density functions in continuous domains. A hybrid ant colony optimization (HACO) incorporating with continuous population-based incremental learning (PBILc) and differential evolution (DE) (Sebag

* Corresponding author. Tel.: +86 20 85211352 401
E-mail address: xiaojing@scnu.edu.cn (J. Xiao).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.02.151

& Ducoulombier, 1998) is proposed in this paper. HACO employs the multi-Gaussian pdfs sampling in parallel and learns the next generation's mean and variance values dynamically by PBILc. The rank-based selection method in ACOR may suffer local optima when the solutions in the archive are very close to each other. To alleviate this, HACO involves a differential evolution operator with linear combination of the one best and two random individuals (Montes and Velazquez-Reyes, 2006) in the current population. The combination of differential evolution and ant colony has been applied to feature selection and power systems (Chiou, Chang, & Su, 2004; Khushaba, Al-Ai, Alsukker, & Al-Jumaily, 2008).

The rest of this paper is organized as follows. Section 2 describes HACO in detail. The pheromone representation and Gaussian *pdf* generation are elaborated in Section 2. Experimental results and analysis are presented in Section 3. Section 4 discusses how two different DE operators affect HACO. We conclude this paper and outline the future works in Section 5.

2. HACO: A hybrid ant colony optimization for continuous domains

In this section, we introduce the HACO in detail. First, we describe the pheromone representation in HACO and the overall framework of the algorithm is given then. The solution construction and Gaussian *pdf* generation by two methods are introduced finally.

2.1. Pheromone representation

In traditional ACO algorithms, ants use pheromone to communicate with each other. In combinatorial optimization problem, ACO algorithms use a table to store pheromone information. At each construction step, an ant uses the table to form a discrete probability distribution. While in continuous optimization, different representation should be adopted.

In this paper, we use the same idea to that proposed by Marco Dorigo in ACOR (Socha & Dorigo, 2008). HACO stores a number of solutions in a solution archive. Each solution contains the values of its n variables. The solution archive stores the solutions and also their values of the objective functions.

As the pheromone information is not explicitly stored as in traditional ACO, pheromone cannot be updated directly. In HACO, pheromone update is accomplished by updating the solution archive. When a better solution is found, it will be added to the solution archive with a worst solution being removed. With the best solutions ever found keeping in the solution archive, the ants can explore the domain effectively.

2.2. HACO framework

The framework of HACO algorithm is described in Algorithm 1 and the notations of the algorithm are presented in Table 1.

Table 1
Notations in HACO.

Symbol	Description
$X = \{x_1, x_2, \ldots, x_n\}$	A set of n continuous variables in a certain problem
k	Size of the solution archive
m	Number of ants used in each iteration
ξ	Speed of convergence
q	Locality of the search process
n	Dimension of the problem
α	Learning rate
$N(\mu, \sigma)$	Gaussian function with mean μ and standard deviation σ
s	Solution in the solution archive
F	Coefficient in differential evolution

Algorithm 1. HACO Algorithm.

```
set iteration t = 0; initialization_of_k_solutions();
while t < maxNumIterations do
  t = t + 1;
  for j = 1 to m/2 do
    for each dimension i do
      sampling the value of dimension i by the Gaussian
  kernel pdf by ranking;
  for j = m/2 + 1 to m do
    for each dimension i do
    sampling the value of dimension i by the Gaussian pdf by
  PBILc;
  for each ant do
    update_solution_archive();
    generate_Gaussian_functions();
```

For an n-dimension problem, an ant constructs a solution in n steps. At each step i, an ant samples a value for variable x_i. HACO keeps k solutions in the archive for pheromone update calculation. The ith variable of jth solution is denoted by s_i^j. *intialization_of_k_solutions*() initially generate k solutions from the scratch by uniform sampling. *update_solution_archive*() means that the pheromone update is accomplished by adding the set of newly generated better solutions according to the ranking or PBILc method, then the same number of worst solutions in the archive is removed accordingly. In HACO *generate_Gaussian_functions*() performs two kinds of Gaussian generation. One of the ant groups applies the rank-based selection mechanism as in ACO_R and the other applies the PBILc method which will be introduced later on. Then the two ant groups use the corresponding Gaussian functions to sample their values for each dimension.

2.3. Generate Gaussian functions by rank-based scheme

For each dimension i, HACO generates a probability density function called Gaussian kernel *pdf*. First, each solution constructs an individual Gaussian *pdf* $Ni(\mu_i, \sigma_i)$ for each dimension. Then the Gaussian kernel *pdf* is making up by the k separate Gaussian functions with different weights. These weights are associated with the fitness value of the objective function. Better solution will have higher weight. The weight w_l of the solution s^l is calculated by the following equation (Socha & Dorigo, 2008):

$$w_l = \frac{1}{qk\sqrt{2\pi}} e^{-\frac{(l-1)^2}{2q^2k^2}}, \tag{1}$$

which defines the weight to be a value of the Gaussian function with argument l, mean 1.0, and standard deviation qk, where q is a parameter to balance between diversification and intensification. When q is small, the best-ranked solutions are strongly preferred, and when it is large, the probability becomes uniform (Socha & Dorigo, 2008).

Sampling the Gaussian kernel *pdf* is done in two phases. First, choose one of the individual Gaussian function that compose the Gaussian kernel with probability p_l given by Eq. (2). Then sample the chosen Gaussian function.

$$p_l = \frac{w_l}{\sum_{r=1}^{k} w_r}. \tag{2}$$

We consider the chosen Gaussian function by (2) with μ_i a standard solution for other ant solutions to explore. To establish the value of the standard deviation σ_i at step i, we calculate the average distance from μ_i to all solutions in the archive, and multiply it by the parameter ξ: